System design: Newsfeed System

Suresh Podeti

5 min read · Oct 24, 2023

Introduction

All the social media sites have some sort of news feed system, like those in Facebook, Twitter, Instagram, Quora, and Medium.

News feed is a list of posts with text, image, or video generated by other entities in the system tailored for you to read. It's constantly updating while other entities creating new posts.

The challenging task is to provide a personalized newsfeed in real-time while keeping the system scalable and highly available.

Requirements

Functional

- News feed is generated using the posts from other entities in the system that the user followed or the user might be interested in.
- Posts might have text, image and video.
- The new posts generated by others should be appended to the news feed of the user based on some ranking mechanism.

Non-functional

- **Scalability:** Our proposed system should be highly scalable to support the ever-increasing number of users on any platform, such as Twitter, Facebook, and Instagram.
- **Fault tolerance:** As the system should be handling a large amount of data; therefore, partition tolerance (system availability in the events of network

failure between the system's components) is necessary.

- **Availability:** The service must be highly available to keep the users engaged with the platform. The system can compromise strong consistency for availability and fault tolerance, according to the PACELC theorem.
- **Low latency:** The system should provide newsfeeds in real-time. Hence, the maximum latency should not be greater than 2 seconds.

Database Design

Entities

- **User** — This relation contains data about a user. A user can also be a follower or friend of other users.
- **Entity** (page, group, etc): entityId, name, description, timestamp
- **Feed_Item**: id, title, text, authorId, timestamp
- **Media**: mediaId, url, timestamp

Relationships

- Follower-Followee: A User can follow other Users or Entities. (m:n)
- Author-Feed_Item: Users and Entities can generate feed item. For simplicity, assume only Users can generate feed items. (1:n; we can embed the authorId)
- Feed_Item-Media: Each feed item has some associated Medias. (1:n)

We use a **graph database** to store relationships between users, friends, and followers.

User	
User_ID:	varchar(32) (PK)
Name:	varchar(32)
Email:	varchar (32)
CreationDate:	datetime
Mobile:	varchar (32)
LastLogin:	datetime

Entity (Page or Group)	
Entity_ID:	varchar (PK)
Name:	varchar(32)
Description:	varchar(512)
CreationDate:	datetime
Creator:	User_ID (FK)

Feed_Item	
Feed_Item_ID:	varchar (PK)
Creator:	User_ID(FK)
Content:	varchar(512)
Entity_ID:	Entity_ID (FK)
CreationDate:	datetime
Likes_count:	int
Media_ID:	int

Media	
Media_ID:	int (PK)
Desciption:	varchar (256)
Path:	varchar(256)
Views_count:	int
CreationDate:	datetime

Fig 1.0: The database schema for the newsfeed system

High level design

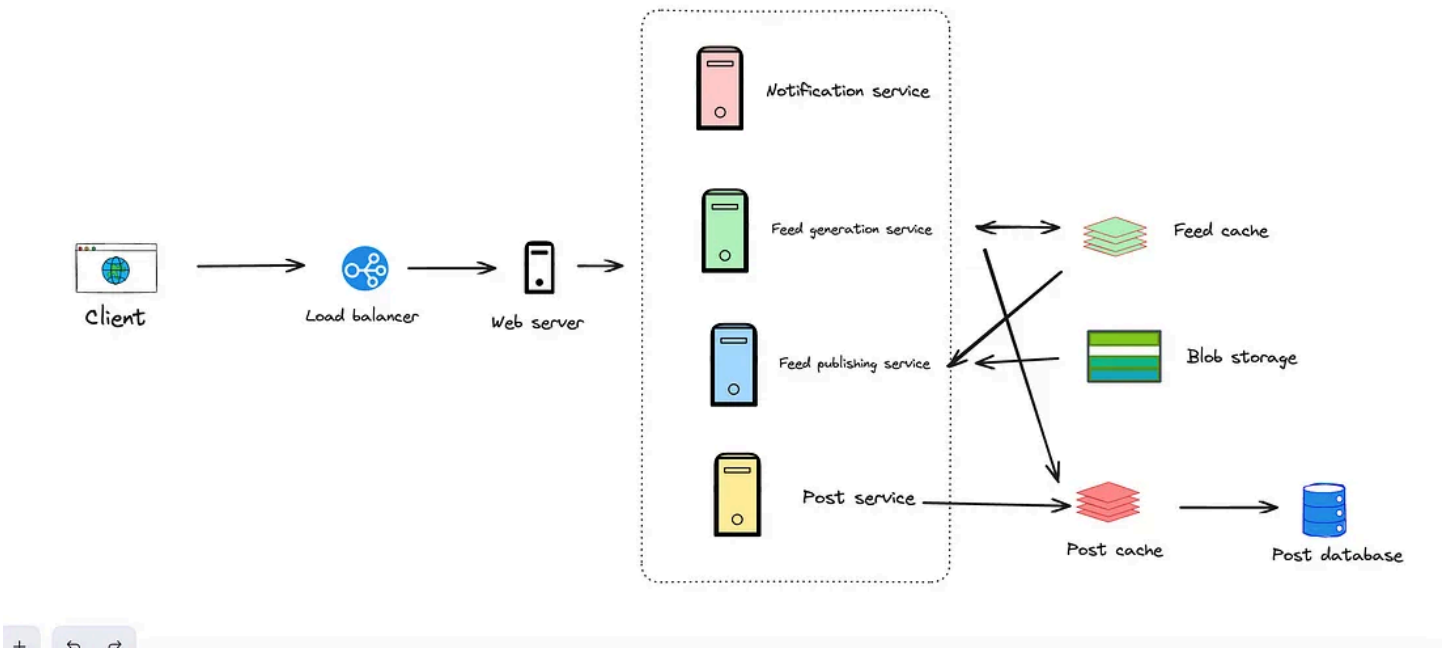

Fig 2.0: High level design of the Newsfeed system

- **Post-service:** Whenever a user requests to create a post, the post-service is called, and the created post is stored on the post database and corresponding cache. The media content in the post is stored in the blob storage.
- **Notification service:** It informs the newsfeed generation service whenever a new post is available from one's friends or followers, and sends a push notification.
- **Newsfeed generation service:** This service generates newsfeeds from the posts of followers/friends of a user and keeps them in the newsfeed cache.
- **Newsfeed publishing service:** This service is responsible for publishing newsfeeds to a users' timeline from the newsfeed cache. It also appends a thumbnail of the media content from the blob storage and its link to the newsfeed intended for a user.

Detailed design

The newsfeed generation service

Newsfeed is generated by aggregated posts (or feed items) from the user's friends, followers, and other entities (pages and groups).

In our proposed design, the **newsfeed generation service** is responsible for generating the newsfeed. When a request from a user to retrieve a newsfeed is received at web servers, the web server either:

- Calls the newsfeed generation service to generate feeds because some users don't often visit the platform, so their feeds are **generated on their request**.
- It fetches the **pre-generated** newsfeed for active users who visit the platform frequently.

The following steps are performed in sequence to generate a newsfeed for Alice:

1. The newsfeed generation service **retrieves IDs** of all users and entities that Alice follows from the **graph database**.
2. When the IDs are retrieved from the graph database, the next step is to get their friends' (followers and entities) information from the user cache, which is regularly updated whenever the **users database** gets updated/modified.
3. In this step, the service **retrieves** the **latest**, most popular, and relevant **posts for those IDs from the post cache**. These are the posts that we might be able to display on Alice's newsfeed.
4. The **ranking service** ranks posts based on their relevance to Alice. This represents Alice's current newsfeed.
5. The newsfeed is **stored in the the newsfeed cache** from which top *N* posts are published to Alice's timeline.
6. In the end, whenever Alice reaches the end of her timeline, the next top *N* posts are fetched to her screen from the newsfeed cache.

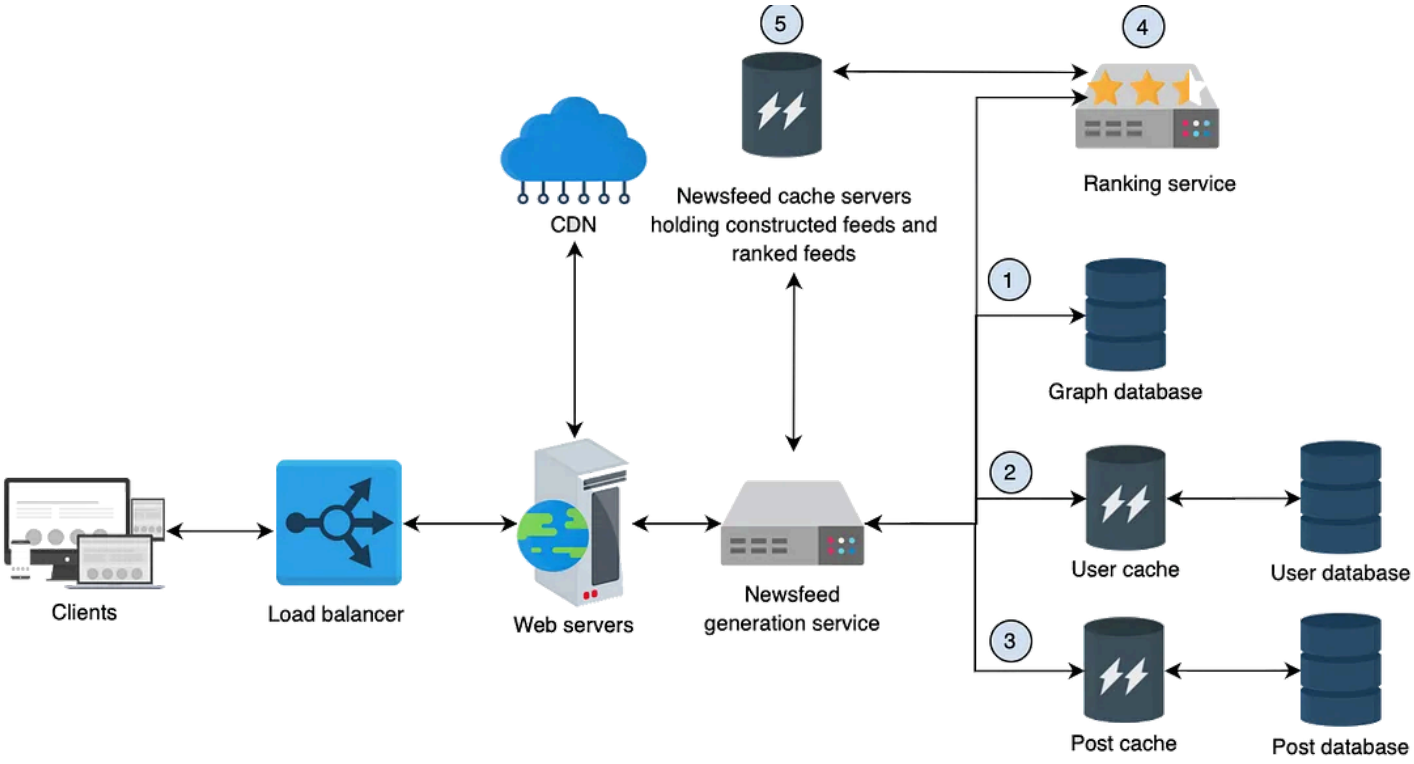

Fig 3.0: Working of the newsfeed generation service

The newsfeed publishing service

At this stage, the newsfeeds are generated for users from their respective friends, followers, and entities and are stored in the form of `<Post_ID, User_ID>` in the news feed cache.

Now the question is how the newsfeeds generated for Alice will be published to his timeline?

The **newsfeed publishing service** fetches a list of post IDs from the newsfeed cache. The data fetched from the newsfeed cache is a tuple of post and user IDs, that is., `<Post_ID, User_ID>`. Therefore, the complete data about posts and users are retrieved from the users and posts cache to create an entirely constructed newsfeed.

In the last step, the fully constructed newsfeed is sent to the client (Alice) using one of the **fan-out approaches**. The popular newsfeed and media content are also stored in CDN for fast retrieval.

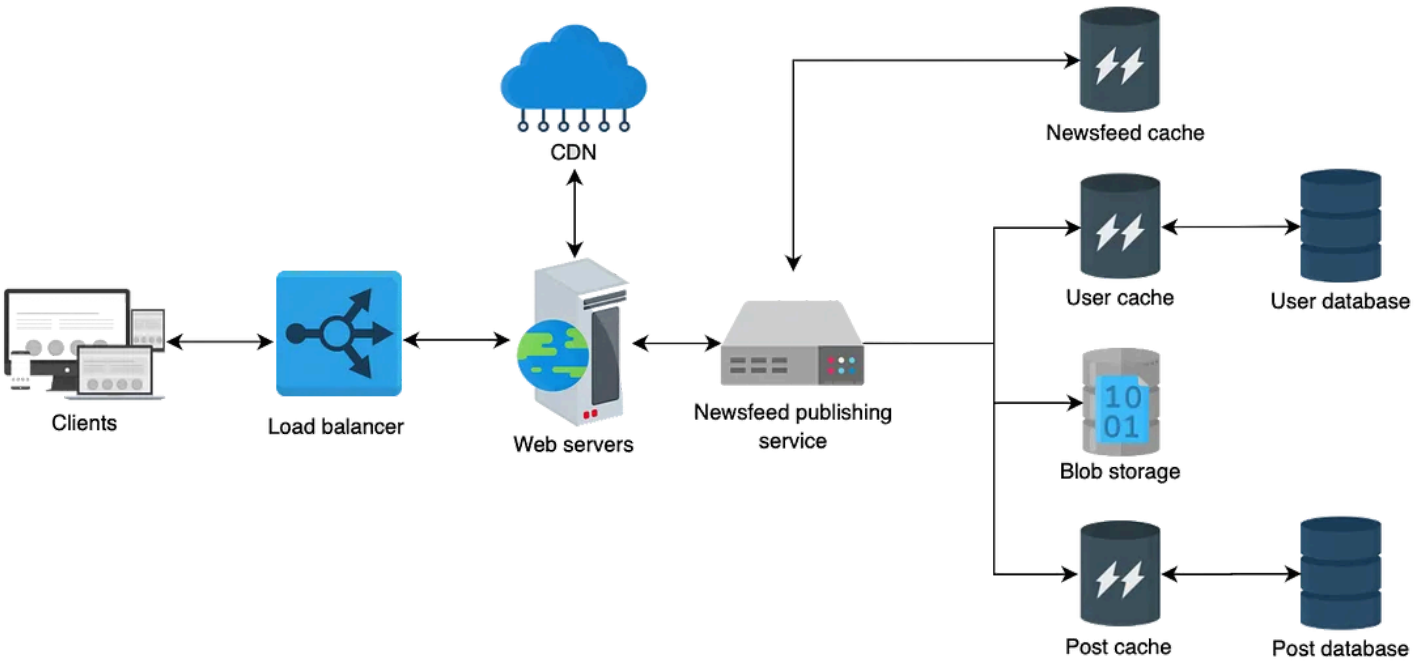

Fig 4.0: The newsfeed publishing service in action

The newsfeed ranking service

Social media platforms use advanced algorithms to rank and recommend posts on our newsfeeds. Our newsfeed ranking service employs these algorithms, analyzing user history, likes, dislikes, comments, clicks, interactions, and topic preferences. It selects candidate posts, filters out misinformation, identifies frequent interactions, and highlights topics of interest, resulting in a personalized and relevant newsfeed.

Evaluation

- **Scalability:** The proposed system is scalable to handle an ever-increasing number of users. The required resources, including load balancers, web servers, and other relevant servers, are added/removed on demand.

- **Fault tolerance:** The replication of data consisting of users' metadata, posts, and newsfeed makes the system fault-tolerant. Moreover, the redundant resources are always there to handle the failure of a server or its component.

- **Availability:** The system is highly available by providing **redundant servers and replicating data on them**. When a user gets disconnected due to some fault in the server, the session is re-created via a load balancer with a different server. Moreover, the data (users metadata, posts, and newsfeeds) is stored on different and redundant database clusters, which provides high availability and durability.

- **Low latency:** We can minimize the system's latency at various levels by:

- Geographically distributed servers and the cache associated with them. This way, we bring the service close to users.
- Using CDNs for frequently accessed newsfeeds and media content.

News Feed Ranking | System Design Interview | Software Architect | Software Architecture | Distributed Systems

Edit profile

Written by Suresh Podeti

1.2K Followers

Aspiring Software Architect | SDE III (Golang) @JungleeGames| Ex-Byju's | Co-founder @Rupant Tech. | Ex-Synopsys | I.I.T Kharagpur | 7+ Years of Experience

Recommended from Medium